Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Davor Muc

Razvoj aplikacije na Ethereum verigi blokov

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Luka Šajn

Ljubljana, 2018

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Za potrebe posameznikov ali podjetij, ki se ukvarjajo s projekti, pri katerih za razvoj potrebujejo finančna sredstva in jih želijo zbrati preko tehnologije veriženja blokov. Izdelajte pametne pogo dbe, ki bodo uporabnikom omogočale izdajo svojih žetonov in s tem kopičenje finančnih sredstev. V ta namen podrobno preučite zahteve, izberite primerne tehnologije ter predstavite razvoj in uporabnost rešitve. Aplikacija naj omogoča vlaganje za več uporabnikov ter ustvarjanje in razporeditev žetonov med uporabnike. V zadnjem delu naloge analizirajte opravljeno delo.

Zahvalil bi se svojim staršem in prijateljem, ki so me podpirali med študijem ter mentorju doc. dr. Luki Šajn za vso pomoč pri izdelavi diplomske naloge.

Družini

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ICO	initial coin offering	začetna ponudba kovancev
EVM	Ethereum virtual machine	Ethereum navidezni stroj
DOS	denial of service	zavrnitev storitve
ABI	application binary interface	aplikacijski binarni vmesnik
P2P	peer to peer	enak z enakim
IP	internet protocol	internetni protokol
LIFO	last in first out	nakopičevalni pomnilnik, sklad
IDE	integrated development environment	integrirano razvojno okolje
FRIK	faculty of computer and information science coin	fakulteta za računalništvo in informatiko kovanec

Povzetek

Naslov: Razvoj aplikacije na Ethereum verigi blokov

Avtor: Davor Muc

Problem, ki sem ga obravnaval v diplomski nalogi, je razvijanje sistema za pomoč podjetjem ali posamezniku v zbiranju sredstev za zagon projekta in izdajo svojega žetona na Ethereum verigi blokov. Sistem pomaga podjetjem, ki so se odločila za zbiranje sredstev s pomočjo Ethereum platforme. Za razvoj aplikacije so se uporabile nove tehnologije, ki predstavljajo naslednjo generacijo svetovnega spleta, ki temelji na decentraliziranih aplikacijah. Predstavljen je razvoj potrebnih pametnih pogodb, namestitev le-teh v verigo blokov ter razlaga delovanja aplikacije. Ethereum je decentralizirana platforma za izdelavo in poganjanje pametnih pogodb ter decentraliziranih aplikacij. Pri razvoju sem spisal pametni pogodbi, ki sem ju namestil v omrežje in izvedel celoten postopek pridobivanja sredstev in izdaje žetonov. Razvoj sem začel izvajati na zasebnem omrežju, nadaljeval, kasneje tudi testiral na Ropsten razvojnem omrežju.

Ključne besede: Ethereum, veriga blokov, pametne pogodbe, prodaja žetonov.

Abstract

Title: Developing a crowdsale application on Ethereum blockchain

Author: Davor Muc

The problem that I discussed in the diploma thesis is developing a system to help businesses or individuals collect funds to start the project and issue their token on the Ethereum blockchain. The system helps companies that have decided to raise funds through the Ethereum platform. For the development of the application, new technologies were introduced, representing the next generation of the World Wide Web based on decentralized applications. The development of the necessary smart contracts, the installation of them in the blockchain and the explanation of the operation of the application is presented. Ethereum is a decentralized platform for creating and running smart contracts and decentralized applications. When developing, I wrote two smart contracts that I deployed on the network and performed the entire process of obtaining funds and issuing tokens. I started developing on a private network and later continued, on a Ropsten test network.

Keywords: Ethereum, blockchain, smart contracts, crowdsale.

Uvod

Prvi podatkovni blok je ustvaril Satoshi Nakamoto leta 2008. Že naslednje leto je v tehnologijo veriženja blokov implementiral prvo digitalno valuto, ki jo danes poznamo pod imenom Bitcoin. Če bi tehnologijo veriženja blokov uporabljali le kot metodo, na kateri temelji elektronski denar, ne bi bila tako pomembna. Bistveno večji je potencial njene uporabe predvsem zato, ker mnogo ljudi pričakuje, da bo povzročila revolucijo v načinu uporabe interneta ter poslovanja.

Prihod in razvoj te tehnologije označuje novo dobo svetovnega spleta, zato sem se odločil, da se bom za diplomsko nalogo ob zaključku študija bolj poglobil v delovanje in razvoj aplikacije na enem izmed najbolj naraščajočih projektov z uporabo verige blokov, Ethereum.

Poglavje 1

Osnovne tehnologije veriženja blokov

Tehnologija veriženja blokov (ang. blockchain) je neskončno naraščajoč seznam blokov, ki so med seboj povezani z uporabo kriptografije. Vsak blok tipično vsebuje kazalec, ki je kriptiran z razpršilnim algoritmom in kaže na prejšnji blok, časovno oznako ter transakcijske podatke. Po strukturi so podatkovni bloki odporni na zlonamerne vdore, ki bi lahko spremenili informacije v samem bloku. Distribuiranje blokov se vrši preko P2P (ang. peer-to-peer) omrežja, ki obenem skrbi tudi za zgradbo novih blokov[10].

Podatkovni bloki so decentralizirani, informacije v njih so zapisane permanentno. Ko se blok dokončno sformira, informacij na njem ni več mogoče spreminjati. Zato je tehnologija veriženja blokov zelo primerna za hranjenje zdravstvenih podatkov, beleženje pomembnih dogodkov, identitete posameznikov, beleženje transakcij itd.

Idejo, ki problem z zaupanjem učinkovito razreši, je oktobra leta 2008 predlagal Satoshi Nakamoto. Ime, ki predstavlja to idejo, je tehnologija veriženja blokov oziroma sistem razpršenega veriženja blokov. Omogoča varno in učinkovito upravljanje seznama lastnikov, nečesa digitalnega, ne

da bi za to potrebovali posrednika, ki bi mu morali vsi zaupati.

Kako strukturno preprečiti nekontrolirano kopiranje ter goljufanje elektronskega denarja, temelji na dokazu opravljenega dela. Ta približno vsakih 10 minut zahteva sprejetje novega bloka oziroma seznama na novo opravljenih transakcij. Z večanjem opravljenega dela se veča tudi delo potencialnega nepridiprava, saj bi moral opraviti bistveno več dela kot vsi drugi skrbniki sistema skupaj, kar je v praksi težko izvedljivo.

Protokol usklajevanja temelji na prikazu opravljenega dela in izhaja iz izvajanja zapletenih računskih problemov. Ko nekdo takšno nalogo razreši, je mogoče rešitev preprosto preveriti in potrditi, da je bilo delo opravljeno in ni prišlo do prevare[10]. Tako rekoč bi lahko protokol primerjali kar z izgradnjo zapletene sestavljanke z veliko majhnih delcev. Ko je sestavljena, lahko z enim samim pogledom nanjo preverimo, ali je bilo delo res skrbno opravljeno. Prikaz rešitve je nesporen dokaz, da je bilo v reševanje investiranega veliko časa in energije.

1.1 Pametne pogodbe

Res je, da je pri sami tehnologiji veriženja blokov najbolj razširjen digitalni denar, vendar nam sama tehnologija omogoča tudi digitalno sestavljanje pogodb, različnih listin in potrdil, podatke o sledljivosti hrane, zdravil, rudnin in podobno.

Vsi si zelo veliko obetajo tudi od pametnih pogodb, ki so shranjene v verigi blokov in se same izvršujejo, kadar so izpolnjeni v njih vsi vgrajeni pogoji[10]. Denimo da se dogovorimo, da nam pripada določen odstotek dohodka od vsakega prodanega izdelka, pri nastajanju katerega smo sodelovali. Pametna pogodba nam omogoči, da se dogovor uresničuje kar sproti. Torej ko pride do transakcije, ki ustreza pogojem iz pogodbe, se delež denarja, ki

nam po pogodbi pripada, avtomatsko nakaže na naš račun. Pogodba je zapisana v verigi, ki je shranjena pri vseh uporabnikih sistema, zatorej izvajanja dogovora ne more nihče preprečiti.

Poglavje 2

Ethereum

Namen Ethereuma je ustvariti alternativni protokol za razvoj decentralizirane aplikacije. Lahko si ga predstavljamo kot velik svetovni računalnik, ki ima svojo verigo blokov z vgrajenim »Turing-complete«[7] programskim jezikom, ki omogoča vsem, da lahko sami pišejo tako imenovane pametne pogodbe ter decentralizirane aplikacije, kjer lahko svobodno ustvarjajo svoja pravila, transakcijske formate in funkcije. Pametne pogodbe, ki vsebujejo vrednosti in se odklenejo le, kadar so določeni pogoji izpolnjeni, so glavna razlika med Bitcoin skriptnim jezikom. Vse to zaradi zavedanja vrednosti in stanja v verigi blokov.

2.1 Ethereum računi

V Ethereumu je stanje bloka hranjeno v objektih imenovani računi. Vsak račun ima 20 bajtov dolg naslov. Ethereum račun vsebuje štiri polja:

- števec, ki služi za kontrolo, da se vsaka transakcija izvede le enkrat,
- trenutno stanje na računu,
- koda pametne pogodbe na računu,
- shramba računa (privzeto je prazna).

Ether je osnovna interna kripto valuta za plačevanje pristojbine za transakcije med računi. Obstajata dva tipa računov. Prvi je račun v zunanji lasti, ki je kontroliran s strani zasebnega ključa. Drugi je pogodbeni račun, ki jih nadzira njihova pogodbena številka. Račun v zunanji lasti ne vsebuje kode, kar pomeni, da lahko pošiljamo sporočila z ustvarjanjem in podpisovanjem transakcije. V pogodbenem računu se koda aktivira ob prejetju sporočila, ki omogoča ustvarjanje novih pogodb, pošiljanje drugih sporočil ter branje in pisanje v notranjem pomnilniku.[7]

Pogodbe so v Ethereumu neke vrste "avtonomi agenti", ki živijo v Ethereumovem okolju in izvajajo določen kos kode, kadar ga sporočilo ali transakcija sproži. Imajo neposreden nadzor nad stanjem Ethera na računu in lastno zalogo ključev ali vrednosti.

2.2 Transkacije in sporočila

2.2.1 Transakcije

Izraz transakcija se v Ethereumu uporablja za sklicevanje na podpisan paket podatkov, ki vsebuje sporočilo poslano z računa v zunanji lasti.

Transakcija vsebuje:

- prejemnika sporočila,
- podpis, ki identificira pošiljatelja,
- količino Ethera za prenos od pošiljatelja do prejemnika,
- neobvezno podatkovno polje,
- vrednost STARTGAS, ki predstavlja največje število računskih korakov, ki jih lahko izvrši transakcija,

- vrednost GASPRICE, ki predstavlja pristojbino, ki jo pošiljatelj plača za vsak računski korak.

Prve tri alineje predstavljajo standardna polja, pričakovana v katerikoli kripto valuti. Podatkovno polje nima privzete funkcije, ampak navidezni stroj, ki ima operacijsko kodo, s katero lahko pogodba dostopa do podatkov.

Polji STARTGAS in GASPRICE sta ključnega pomena za Ethereumov anti-DOS model[7]. Da bi preprečili nenamerne ali zlonamerne neskončne zanke ali druge računalniške izgube v kodi, mora vsaka transakcija določiti omejitev, koliko računskih korakov izvajanja kode lahko uporabi. Temeljna enota za izračun je gas. Običajno en računalniški korak stane en gas, vendar pa nekateri postopki stanejo več zato ker so računsko dražji ali povečajo količino podatkov, ki jih je treba shraniti kot del stanja. Obstaja tudi pristojbina pet gasov enot za vsak bajt v transakcijskih podatkih. Namen sistema s pristojbino je, da zahteva od napadalca, da za vsak vir, ki ga porabi, plača sorazmerno, vključno z računanjem, pasovno širino in pomnilnikom. Zato mora vsaka transakcija, ki porabi več računske moči, sorazmerno več plačati.

2.2.2 Sporočila

Pogodbe so zmožne pošiljati sporočila drugim pogodbam. Sporočila so navidezni objekti, ki niso nikoli izvedena zaporedno in obstajajo le v Ethereum okolju[7].

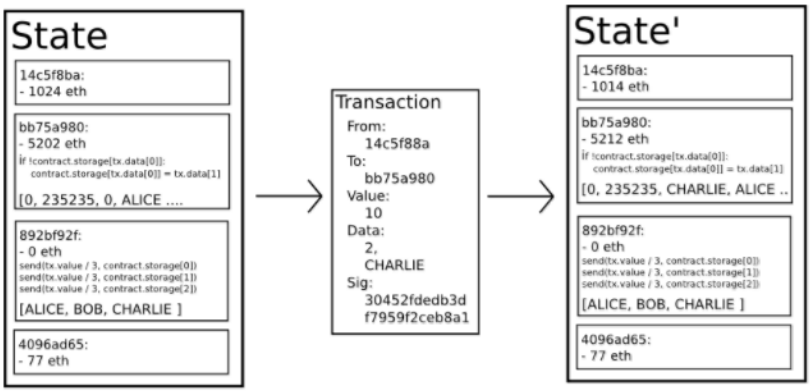
Sporočilo vsebuje:

- pošiljatelja sporočila,
- prejemnika,
- količina Ethera za prenos ob sporočilu,
- neobvezno podatkovno polje,

- vrednost STARTGAS.

V bistvu je sporočilo enako kot transakcija, z izjemo, da ga proizvede pogodba in ne zunanji udeleženec. Sporočilo se sproducira, ko pogodba, ki se trenutno izvaja, sproži operacijsko kodo CALL. Ta ustvari in izvede sporočilo. Kot transakcija se tudi sporočilo pošilja do prejemnika, ki izvaja svojo kodo. Tako imajo pogodbe popolnoma enak odnos kot zunanji akterji.

2.3 Funkcija prehajanja stanja



Slika 2.1: Funkcija prehajanja stanja [7]

Funkcijo prehoda stanja Ethereum definiramo, tako:

1. preverimo, ali je transakcija dobro formatirana (ima pravilno število vrednosti), če je podpis veljaven ter če se števec pošiljatelja in prejemnika ujemata. Če ne, vrnemo napako;

2. izračunamo provizijo za transakcijo po enačbi STARTGAS * GASPRICE in določimo naslov pošiljatelja. Odštejemo pristojbino s plačilnega računa pošiljatelja in povečamo pošiljateljev števec. Vrnemo napako, če ni dovolj za plačilo;

3. inicializiramo GAS = STARTGAS in odvzamemo določeno količino gasa enot za vsak bajt, da lahko plačamo transakcijo;

4. prenesemo vrednost transakcije s pošiljateljevega računa v račun prejemnika. Če račun za prejemanje še ne obstaja, ga ustvarimo. Njegovo kodo poganjamo do konca ali dokler ne ostanemo brez gasa, če je prejemni račun pogodba;

5. ob neuspešnem prenosu vrednosti, morda, ker pošiljatelj ni imel dovolj denarja ali pa je za izvedbo kode zmanjkalo gasa, povrnemo vse prehode stanja, razen plačila pristojbin, v nasprotnem primeru ves denar za plačilo pristojbin vrnemo pošiljatelju.

2.4 Izvajanje kode

Koda v Ethereum pogodbah je napisana v jeziku nižjega nivoja, ki temelji na bajtnem jeziku in se imenuje EVM-koda. Sestavljena je iz niza bajtov, kjer vsak predstavlja operacijo. Izvedba kode je neskončna zanka, ki je sestavljena iz večkratnega izvajanja operacije na trenutnem števcu programa. Začne z nič, nato povečuje števec za ena, dokler ne doseže konca kode, napake ali ukazov STOP in RETURN[6].
Operacije imajo dostop do treh vrst prostora, v katerih lahko shranjujejo podatke:

- sklad, LIFO vsebovalnik, v katerega lahko potisnemo in izpuščamo vrednosti (ang. push in pop);
- pomnilnik, neskončno naraščajoča bajtna tabela;
- dolgoročna shramba v pogodbi (ključ/vrednost), ki za razliko od sklada in pomnilnika, katerim se vrednosti poenostavijo po izteku računa, v shrambi hranimo vrednosti dolgoročno.

Koda lahko dostopa do vrednosti, pošiljatelja, podatkov dohodnega sporočila in podatkov o glavi bloka. Lahko vrne bajtni niz podatkov kot izhod (ang.

output). Medtem ko teče Ethereumov navidezni stroj, lahko njegovo polno računsko stanje definira »tuple« (stanje bloka, transakcija, sporočilo, koda, pomnilnik, sklad, števec in gasa), kjer je stanje bloka globalno stanje, ki vsebuje vse račune in vključuje ravnovesje in pomnilnik. Na začetku vsakega kroga izvedbe se naslednji ukaz najde tako, da vzamemo bajt kode, na katerega kaže števec. Vsak ukaz ima svoje definicije in pogoje, kako vpliva na »tuple«. Osnovno implementacijo Ethereuma je možno napisati v par sto vrstic kode.

2.5 Veriga blokov in rudarjenje

Ethereumova in Bitcoinova veriga blokov sta si zelo podobni. Glavna razlika je, da Ethereum bloki vsebujejo kopijo seznama transakcij in kopijo zadnjega stanja bloka. Bloki še hranijo številko bloka in težavnost. Osnovni validacijski algoritem v Ethereumu poteka tako:

1. pogleda, ali obstaja prejšnji blok ter preveri njegovo veljavnost;
2. preveri, ali je časovni žig bloka večji od prejšnjega bloka in manjši kot 15 minut od tega trenutka (15 minut v prihodnost);
3. preveri, ali so številka bloka, težavnost, koren transakcije in omejitev gasa veljavne;
4. preveri veljavnost opravljenega dela na bloku;
5. če katerakoli aplikacija vrne napako za transakcijo oziroma, če se je porabila vsa vrednost za plačilo pristojbin, vrne napako

Pristop se lahko zdi zelo neučinkovit na prvi pogled, ker mora shraniti celotno stanje z vsakim blokom, ampak je učinkovitost primerljiva z Bitcoin verigo. Razlog tega je, ker je stanje shranjeno v drevesni strukturi, in za vsakim blokom se mora spremeniti le majhen delež drevesa. Podatki so shranjeni enkrat in dostopani s pomočjo več kazalcev (ang. hashes of subtrees). Za to

poskrbi posebno drevo, imenovano »Patricia tree«, vključno z modifikacijo Merkel drevesnega koncepta, ki omogoča, da so vozlišča efektivno vstavljena, izbrisana in spremenjena. Ker so vse informacije o stanju del zadnjega bloka, ni treba shranjevati celotne zgodovine[6].

Proces izvajanja kode pogodbe je del definicije funkcije prehoda stanja, ki je del validacijskega algoritma bloka. Torej, če se transakcija doda v blok B, se bo koda, ki se je sprožila s to transakcijo, izvedla na vseh vozliščih zdaj in v prihodnosti, ko bi še enkrat prenesli in validirali blok B.

2.5.1 Rudarjenje

Rudarji so računalniki, ki so povezani v Ethereum omrežje, imajo sinhronizirano vozlišče in zagnano rudarjenje. Skupaj zelo hitro ugotavljajo pravilen odgovor sestavljanke, dokler je eden izmed njih ne razreši. Ko rudar najde potrebno vrednost, vsi ostali prenehajo z rudarjenjem na tem bloku. Natančneje, rudarji skozi razpršilno funkcijo poženejo edinstvene meta podatke in zamenjajo »nonce« vrednost, kar vpliva na končni rezultat funkcije. Če rudar pride do rezultata razpršilne funkcije, ki se ujema s trenutnim ciljem, se mu dodeli Ether in odda blok po omrežju. Ob tem ga vsako vozlišče potrdi in doda v svojo kopijo knjige

Nov blok se ustvari približno vsakih 12 do 15 sekund. Če se hitrost ustvarjanja blokov poveča ali zmanjša, algoritem samodejno poveča ali zmanjša težavnost uganke. Rudarji naključno zaslužijo Ether in njihova dobičkonosnost je odvisna od sreče in količine računalniške moči, ki jo namenjajo rudarjenju. Algoritem, ki ga uporablja Ethereum se imenuje »ethash«[12].

2.6 Možne aplikacije na Ethereum platformi

Na Ethereumu so na voljo tri vrste aplikacij. V prvo kategorijo sodijo finančne aplikacije. Te uporabnikom omogočajo uporabo boljših ter močnejših načinov

upravljanja in sklepanja pogodb z uporabo svojega denarja. Vključene so podvalute, pogodbe o varovanju pred tveganjem, varčevalne denarnice, oporoke in tudi zaposlitvene pogodbe. Polfinančna uporaba spada v drugo kategorijo. Pri takšnih aplikacijah gre za mešanico denarne in nedenarne pogodbe. Tretja vrsta so aplikacije, ki niso niti malo finančne obarvane, npr. so spletno glasovanje in decentralizirano državno upravljanje.[10]

Naštetih je nekaj primerov aplikacij:

- Upravljanje identitete. Vsak uporabnik na Ethereum platformi digitalno podpiše vse svoje interakcije ali s pametnimi pogodbami ali z drugimi uporabniki. Zato je mogoče povezati identiteto uporabnika in vse akcije, ki jih bo ta izvedel.

- Zaupnost. Če se sporazum lahko izrazi s programskim jezikom, lahko pametne pogodbe uporabljajo kot način zamenjave (ali uveljavljanja) običajnih pogodb med strankami. Takšne pogodbe je prav tako mogoče uporabljati za pregled vseh transakcij. Recimo, da želi uporabnik donirati svoj denar neprofitni organizaciji in ga zanima, kam ta denar tudi gre. Enostavno lahko napiše aplikacijo, ki v realnem času spremlja poteka denarja.

- Množično financiranje. Ethereumove pametne pogodbe omogočajo ustvarjanje svojih žetonov, ki jih lahko prodamo za pravi denar in s tem financiramo svoje projekte.

- Spletne trgovine, kjer je vsak izdelek zapisan v Ethereumovi verigi blokov in uporabnik tako lahko preveri avtentičnost tega izdelka.

- Lastniški repozitorij. Po končani transakciji v Ethereum omrežju, se informacije, ki smo jih uporabili pri sami transakciji, za vedno zapišejo v verigo blokov. Ti so nespremenljivi, imajo časovni žig, dobijo ga ob dodajanju v verigo in jih ni možno izbrisati. Zaradi teh lastnosti lahko na verigi hranimo dokaz lastništva, registriramo svoj IP, pravice

za pesem ali knjigo ... Vedno lahko dokažemo, kdo je avtor/lastnik dokumenta.

- Državne storitve. Ethereum platforma je primerna za razvoj glasovalnega sistema za državljane.

2.7 Računanje in "Turing-complet" model

Ethereum virtualni stroj deluje na "Turning-complete"principu. To pomeni, da koda na EVM lahko izvaja kakršne koli izračune, ki jih je možno realizirati, vključno z neskončnimi zankami. EVM koda omogoča izvajanje zank na dva načina. Prvič, obstaja navodilo JUMP, ki omogoča programu, da skoči nazaj na prejšnje mesto v kodi, in navodilo JUMPI, ki opravi pogojno skakanje. Drugič, pogodbe lahko kličejo druge pogodbe, ki omogočajo izvajanje zank skozi rekurzijo. To tudi lahko izkoristi napadalec, da lahko preobremeni celotno vozlišče z neskončno zanko. Ta problem se v računalništvu imenuje "halting"[7] problem.

Kot je opisano v sekciji prehajanja stanja v Ethereumu, je to tako rešeno, da vsaka transakcija vsebuje maksimalno število korakov, ki jih lahko naredi. Če za to potrebuje več korakov, se izračun poenostavi, transakcija povrne, vendar se pristojbina vseeno plača. Sporočila delujejo na enak način.

2.8 Valute

Ethereum omrežje vključuje tudi svojo vgrajeno valuto Ether, ki služi za efektivno menjavo med različnimi digitalnimi sredstvi in kar je še pomembnejše, za zagotavljanje mehanizma za plačilo transakcijskih stroškov. Denominacije so označene kot:

- 1: Wei,
- 10^{12}: Szabo,

- 10^{15}: Finney,
- 10^{18}: Ether.

Ether služi za normalne transakcije, Finney za mikro transakcije, Szabo ter Wei naj bi se uporabila za pristojbine in implementacijo protokola[7].

Poglavje 3

Orodja in programski jezik

3.1 Remix IDE

Remix IDE je razvojno orodje in služi kot pripomoček pri programiranju na EVM. Fokusiran je predvsem na razvoj aplikacij, napisanih v programskem jeziku Solidity.

Remix je dobra rešitev za:

- razvoj pametnih pogodb,
- čiščenje napak zagnanih pogodb,
- dostop do stanja in lastnosti že uveljavljene pametne pogodbe,
- odpravljanje napak že izročenih transakcij,
- analizo Solidity kode za zmanjšanje napak v kodi.

3.2 Mist

Mist je denarnica, kjer hranimo Ether. Poleg tega spada tudi med vrste spletnih brskalnikov za decentralizirane aplikacije. Služi za pregled Ethereum pogodb, lahko opravljamo z njimi oziroma jih lahko tudi postavimo

v Ethereum omrežje[2]. Mist zahteva, da ima vsak razvijalec na svojem računalniku, lokalno nameščeno vozlišče. To zavzame precej prostora in obremeni računalnik. Za razvoj na omrežju moramo imeti zagnano generiranje blokov, ki prav tako obremeni procesor. Dobra rešitev je orodje MetaMask.

3.3 MetaMask

MetaMask je vtičnik za Google Chrome brskalnik, ki poenostavi dostop do decentraliziranih aplikacij ali tako imenovanih Dappsov [1]. Ne zahteva, da je na računalniku preneseno celotno Ethereumovo vozlišče. Orodje omogoča uporabniku, da ima shranjenih več identitet, ki razvijalcu omogočajo lažji razvoj ter testiranje aplikacij, saj lahko enostavno sproža in podpisuje transakcije med računi na enem mestu.

3.4 Solidity

Solidity je pogodbeno orientiran programski jezik, ki je namenjen razvoju aplikacij, ki se vršijo na EVM. Na novo razviti jezik temelji ECMAScript sintaksi, ki je zelo podobna Javascriptu[5]. Solidity se prevede v bajtno kodo, ki se izvrši na EVM. Razvijalci v tem programskem jeziku lahko pišejo aplikacije, ki se izvajajo s samoučinkovito poslovno logiko, vsebovano v pametnih pogodbah. Vsi zapisi so nepovratni in verodostojni[8].

3.5 Ropsten

Ropsten je testno omrežje za razvoj pametnih pogodb, preden jih namestimo na glavno omrežje[4]. Na glavnem omrežju je razvoj aplikacij dražji, saj za vsako transakcijo moramo plačati določeno pristojbino. Ob pojavitvi napake v kodi se znesek lahko hitro poveča. Zato se za razvoj uporablja Ropsten omrežje, kjer je vse poenostavljeno in brezplačno. Valuta za transakcije na tem omrežju se imenuje Ropsten Ether.

3.6 Spletna denarnica myetherwallet.com

Myetherwallet je spletna aplikacija, s katero lahko generiramo in dostopamo do svojega naslova v Ethereum verigi blokov[3]. S pomočjo te aplikacije sem namestil pogodbi v omrežje in interaktiral z njimi preko njihovega vmesnika. Napisal sem naslov pogodbe in prilepil ABI-kodo, ki se mi je generirala preko Mist razvojnega okolja. Z myetherwallet sem preverjal in testiral delovanje pametnih pogodb.

Poglavje 4

Razvoj aplikacije za množično prodajo žetonov

4.1 Kaj je začetna ponudba kovancev?

Začetna ponudba kovancev ali ICO (ang. Initial Coin Offering) je sredstvo za množično financiranje, ki se osredotoča na projekte s kriptovalutami. ICO je lahko vir kapitala za zagonska podjetja (ang. Startup). Pri zbiranju denarja za svoj projekt je vlagateljem v zameno za zakonita plačilna sredstva, kot so Bitcoin in Ether, podarjeno določeno število žetonov projekta, ki postanejo funkcionalne valute, če je projekt uspešno financiran.

Za kreiranje začetne ponudbe žetona je treba spisati dve pogodbi, in sicer pogodbo žetona ter pogodbo množične prodaje, ki sta med seboj povezani. Samo slednja lahko izdaja in ustvarja nove žetone. Pogodbi je treba prevesti v bajtno kodo in namestiti na Ropsten omrežje. Običajno za tak projekt je, da ga najprej opišemo in predstavimo delovanje samega projekta na belem papirju (ang. white list).

V svojem diplomskem delu bom prikazal, kako lahko izvedemo množično prodajo žetonov na Ethereum omrežju, da si lahko s pridobljenimi sredstvi

pomagamo pri nadaljnjem razvoju projekta. Kupci žetone dobijo po najnižji ceni in v prihodnosti, ob uspešni izvedbi projekta, mogoče tudi zaslužijo. V nadaljevanju bom opisal postopek izdelave potrebnih pametnih pogodb za pridobivanje finančnih sredstev. Pogodba je namreč razvita na testnem omrežju Ropsten, ker, kot že omenjeno, se na tem omrežju ne uporablja denarja. Testiranje površnih aplikacij bi bilo zelo drago na glavnem omrežju, saj se hitro povečajo stroški, če se recimo pojavi kakšna velika zanka oz. če se zmotimo pri nastavitvi največjega števila korakov.

4.2 Pametna pogodba žetona

Za začetek je bilo treba razviti pametno pogodbo za žeton. V bistvu jo tvorijo tri podpogodbe, in sicer "owned", IERC20Token ter FRIKToken, tako imenovana pogodba žetona, ki razširja prvi dve.

Podpogodba "owned" je sestavljena iz naslova lastnika in metode, ki ga določi. Vsebuje tudi metodo "modifier", s katero lahko lastnik komunicira. Ta se pokaže kot koristna pri metodi "transferOwnershp", ki razširja modifikator, kar pomeni, da lahko lastnik ob klicu na funkcijo spremeni lastnika pogodbe.

V IERC20Token so definirane osnovne funkcije, ki sem jih koristil pri ustvarjanju žetonov. Podpogodba služi kot vmesnik, ki razširja žeton.
Osnovne funkcije so:

- totalSupply - vrača trentno število žetonov, ki so na voljo,
- balanceOf() – ki sprejme naslov lastnika in vrne vrednost, koliko žetonov ima,
- transfer () – služi za prenos sredstev.

Za vsako od zgoraj naštetih funkcij se v datoteko zapiše, kdo je dostopal in kakšni parametri so bili podani v funkcijo. To pripomore k boljšemu pregledu

nad izvajanjem pogodbe.

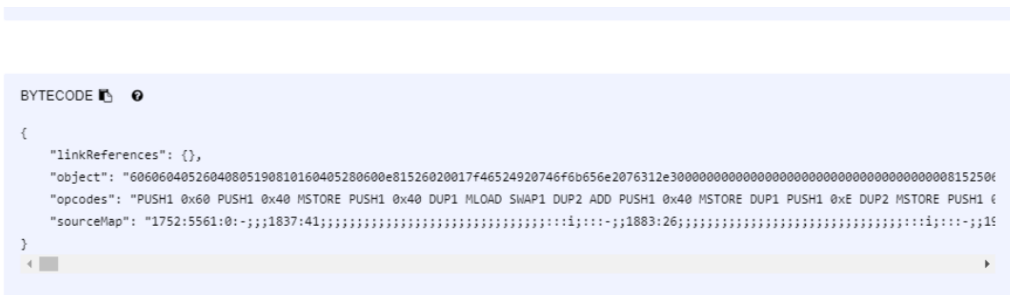
Glavna podpogodba, poimenoval sem jo FRIKToken, razširja in deduje funkcije prej omenjenih podpogodb.

Smiselno sem definiral javne spremenljivke, ki so dostopne vsem:

- string public standard - definira verzijo,
- string public name - definira simbol, ki se bo uporabil pri okrajšvi,
- address public icoContractAddress - naslov iz katerega lahko generiramo nove žetone,
- uint256 supply - spremenljivka, iz katere je vidno, koliko žetonov je bilo generiranih v množični prodaji.

Poleg funkcij, ki so podedovane iz IERC20Token pogodbe, sem dodal še naslednji metodi:

- mintTokens: služi za generiranje žetonov in pošiljanje udeležencu množične prodaje,
- killContract: uporabljamo samo za testiranje pogodbe na testnem omrežju.



Slika 4.1: Bajtna koda pogodbe.

Za vsako funkcijo imam tudi definirane dogodke, ki zapisujejo podatke v blok, da imamo nekakšen pregled nad stanjem pogodbe.

4.3 Pametna pogodba množične prodaje

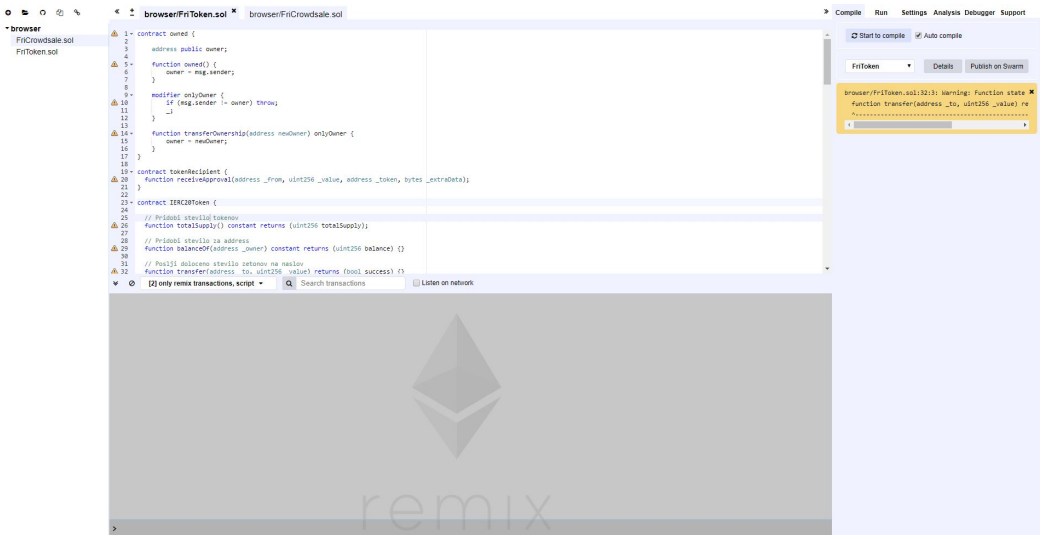
Pogodba FriICO, ki služi za generiranje in pošiljanje žetonov, zajeme naslednje podpogodbe:

- owned - dodeljuje lastnika pogodbe ter ponuja možnost menjave. Enako kot pogodba žetona;
- Pogodba žetona - definira spremenljivke žetona, vsebuje namreč funkcijo za generiranje žetonov in funkcijo za pregled stanja »totalSupply()«;
- Glavna pogodba FriICO.

Glavna pogodba je FriICO, v njej sem definiral spremenljivke:

- startBlock - blok, na katerem se je začela množična prodaja;
- endBlock - označuje konec množične prodaje. Definiral sem ga tako, da sem pogledal, koliko blokov se povprečno ustvari na teden. Dobljeno število sem pomnožil s številom tednov trajanja množične prodaje. Nastavil sem trajanje razprodaje na en mesec;
- minEthToRaise - minimalna vrednost, potrebna pri zbiranju, da lahko prodajo žetonov opredelimo kot uspešno. Če se vrednost ne doseže, se avtomatsko po zaključku pogodbe vsem udeležencem povrne denar;
- maxEtherToRaise - maksimalna vrednost, ki jo lahko zberemo s prodajo. Ko se pride do te vrednosti, se pogodba avtomatično zaključi;
- totalEthRaised - vrednost, kjer beležimo vsoto vseh Etherov, ki so jih udeleženci poslali.

Glavna funkcija v sami pogodbi je zagotovo "payable", ki se izvede, ko uporabnik pošlje Ether na naslov pogodbe. Znotraj te funkcije imam kontrolo nad začetkom množične prodaje ter količino zbranih sredstev, da lahko v primeru dosega maksimalne vrednosti uporabnikom preprečim pošiljanje denarja. Poleg tega kontroliram tudi maksimalni vložek posameznika, da



Slika 4.2: Razvojno okolje Remix IDE.

le-ta ne bi presegel maksimalne vrednosti. Uporabnikov naslov in vrednost Ethera zabeležim v tabelo in povečamo spremenljivko totalEthRaised.

Javna funkcija, dosegljiva zunanjim uporabnikom, je »claimEthIfFailed()«. Uporabnik jo sproži v primeru, ko si želi sam povrniti svoj Ether v že zaključeni prodaji, kjer ni bilo zbranih dovolj sredstev. Funkcija preveri neuspešnost prodaje, sodelovanje uporabnika v prodaji in njegovo povrnitev sredstev.

V pogodbi so opredeljene še funkcije, ki jih lahko sproži le lastnik:

- returnEthIfFailed() – lastnik jo sproži v primeru neuspešne prodaje. Funkcija vsem vlagateljem vrne denar, ki so ga vložili;
- claimTokens() – lastnik jo sproži v primeru, ko imamo določeno število žetonov rezerviranih zase. Funkcija to preveri in izvede transakcijo na naslov, ki je definiran v tabeli »reservedTokens«;
- setTokenContract() – sprejme naslov pogodbe žetona in tako inicializira

spremenljivko FriToken, ki sem jo definiral na začetku.

4.4 Postopek nameščanja na lokalnem okolju

Pri razvoju pogodb sem uporabljal zasebno omrežje (ang. solo network). Proces je enostavnejši, saj ni potrebe po sinhronizaciji celotnega vozlišča na računalniku in tako toliko ne obremenjujemo procesorja. Na slabšem računalniku je proces razvoja pogodbe ter namestitve neposredno preko Mista zelo počasen, zato sem pogodbe razvijal na zasebnem omrežju in kasneje izdelek namestil v Ropsten omrežje. Vendar je za omogočitev treba imeti nameščen Geth. Če torej želimo Mist povezati na zasebno omrežje, moramo v ukazno vrstico vnesti ukaz »geth.exe –dev –ipcpath geth.ipc console«. S tem se začne ustvarjanje novih blokov.

V Mistu je zdaj razvidno, da smo priključeni na zasebno omrežje.

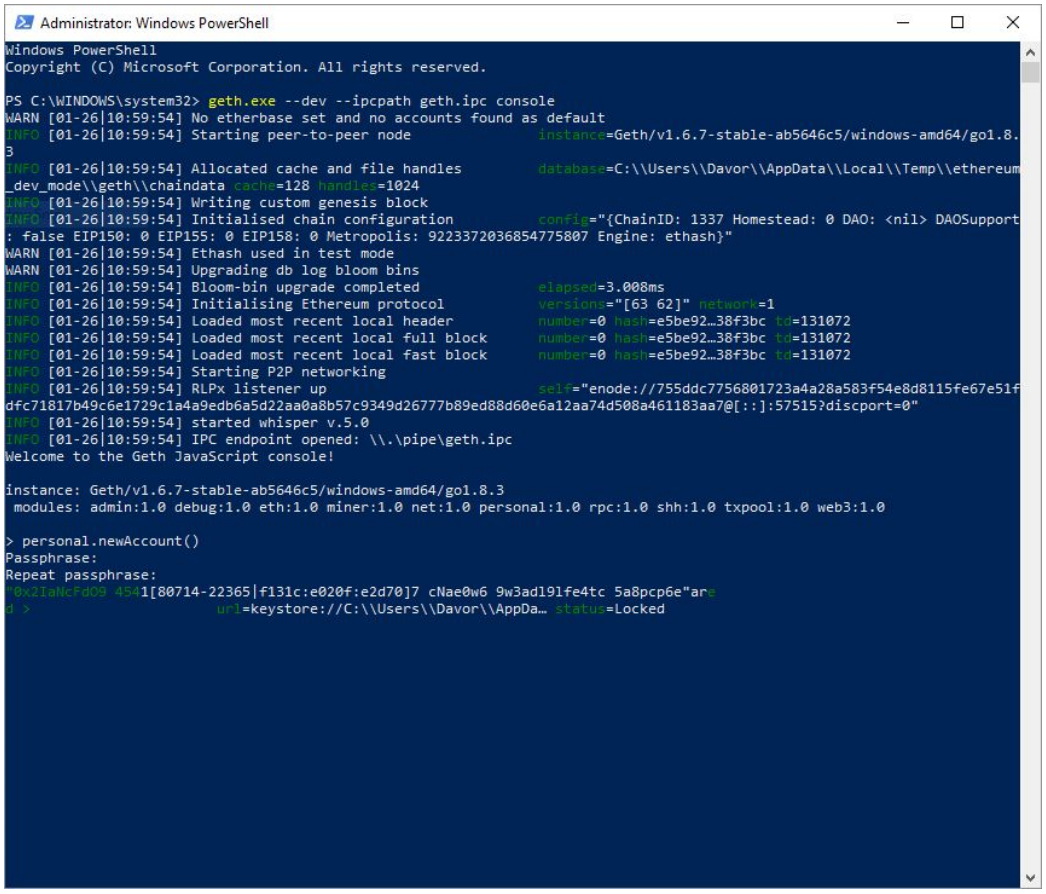
Proces namestitve na verigo blokov je podoben namestitvi na Ropsten omrežje, zato bom opisal namestitev na Ropsten s pomočjo spletne strani www.myetherwallet.com in MetaMask vtičnika. Ob nameščanju preko Mist orodja so nam vse možnosti, ki jih bom opisal v nadaljevanju, na voljo v programu, treba je imeti le sinhronizirano verigo blokov z Ropsten omrežjem.

4.5 Postopek nameščanja na Ropsten omrežje

V brskalniku Chrome sem v MetaMask vtičniku ustvarili račun, preko katerega sem v omrežje namestil pogodbi, ki sem ju razvil v Mist orodju.

Spletna aplikacija www.myetherwallet.com omogoča ogled in komuniciranje s pogodbami, ki so postavljene na omrežju. Najprej je treba določiti omrežje, v katerega bi rad postavili svoje pogodbe, sam sem nastavil na Ropsten omrežje. V zavihku za ogled ali postavitev pogodbe sem v okno vnesel bajtno kodo pogodbe, ki jo kopiram iz razvojnega orodja remixIDE.

Povežem se z enim izmed računov, ki sem jih naredil preko metamaska, in s tem podpišem transakcijo ter postavim na omrežje. Postavim obe po-

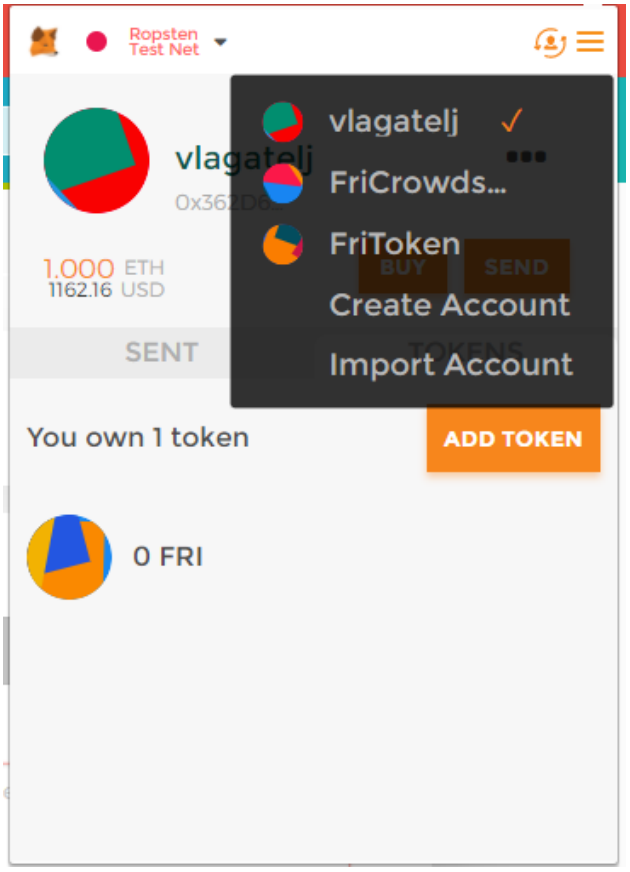


Slika 4.3: Namestitev solo omrežja.

godbi in preko spletne strani »https://ropsten.Etherscan.io«, preverim, če sta pogodbi vidni. Za vsako pogodbo ob postavitvi, sem vnesel koliko gas-a lahko največ porabi pri tem procesu in potrdil transakcijo.

4.6 Delovanje množične prodaje žetonov

Na spletni strani www.myetherwallet.com lahko komuniciramo s pogodbami tako, da v zavihek za ogled pogodb, vnesemo v okno naslov in ABI kodo pogodbe. V spustnem seznamu se pojavijo vse funkcije, ki so definirane v pogodbi. Za vsako se prikažejo različna polja, v katera lahko vnašamo

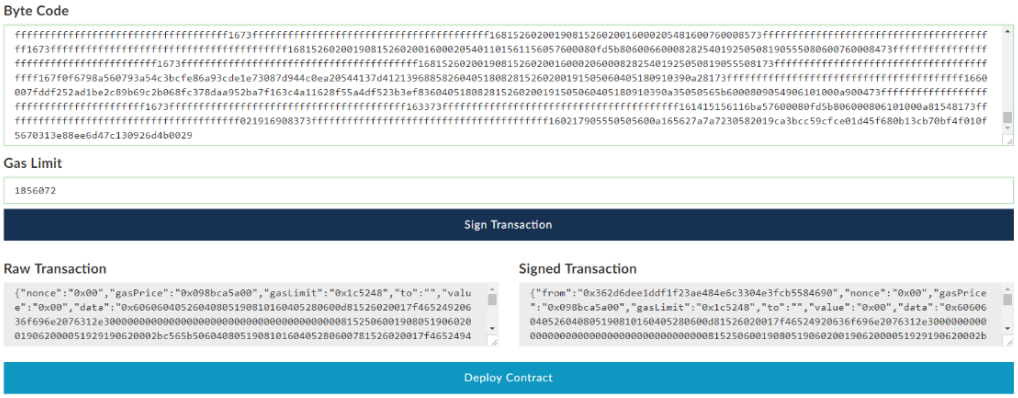


Slika 4.4: Računi na Ethereum omrežju.

vrednosti, ki vračajo rezultate, odvisne seveda od tega, kako so definirane.

Na začetku sem s klicem na funkcijo »setFriTokenAddress()« nastavil naslov pogodb e žetona, ki sem jo ob postavitvi v omrežje dobil, v spletni aplikaciji Etherscan. Ko sem postavil pogodbo, sem s klikom na transkcijo v Metamask dobil naslov, na katerem je postavljena ta pogodba.

Prodaja nastopi, ko se v verigi ustvari številka bloka, ki sem jo predhodno opredelil v pogodbi, traja pa, dokler se ne ustvari zadnji blok. Slednjega izračunam tako, da preverim število že narejenih blokov in s kolikšno hitrostjo se ustvarjajo novi, ter nastavim ustrezne vrednosti. Prodaja se prav



Slika 4.5: Postavitev pogodbe v omrežje.

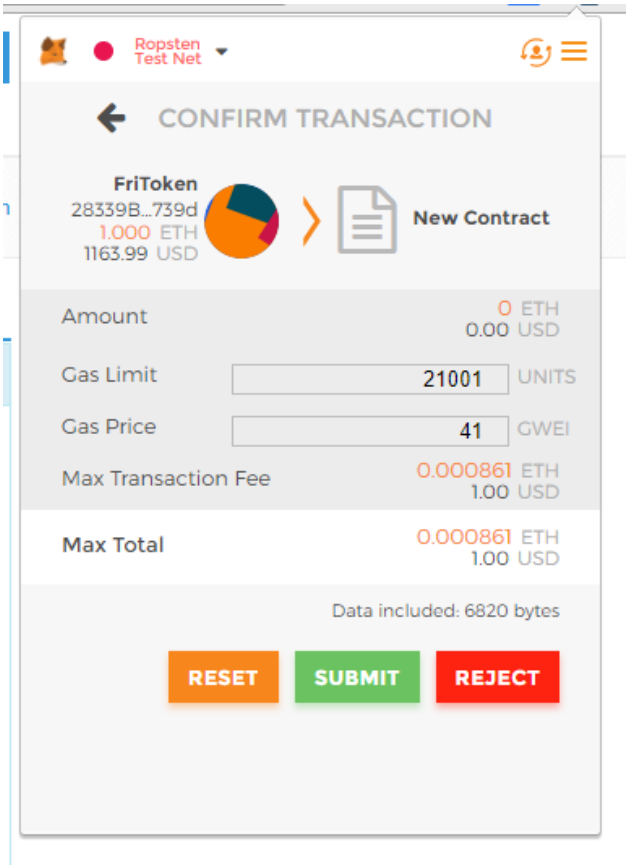
tako zaključi, ko se zbere dovoljšno število sredstev, ki sem jih že vnaprej določi tako, da se želena zbrana vrednost pretvoril v Ether ter nato v Wei in zapisal v pogodbo. Maksimalno vrednost sem nastavil na trideset Etherov in minimalno na tri.

V pogodbi je zapisana funkcija »endCrowdsale()«, s katero preko myehterwallet aplikacije preverim, ali je končana razprodaja. Funkcija »crowdsaleInProgress()« preveri, ali se je prodaja pričela, oz. z drugimi besedami, če se je ustvaril blok, ki sem ga nastavil za začetek prodaje.

S funkcijo »participantCount()« se pregleda število sodelujočih in s funkcijo »contributionInEth()« koliko sredstev je že nabranih.

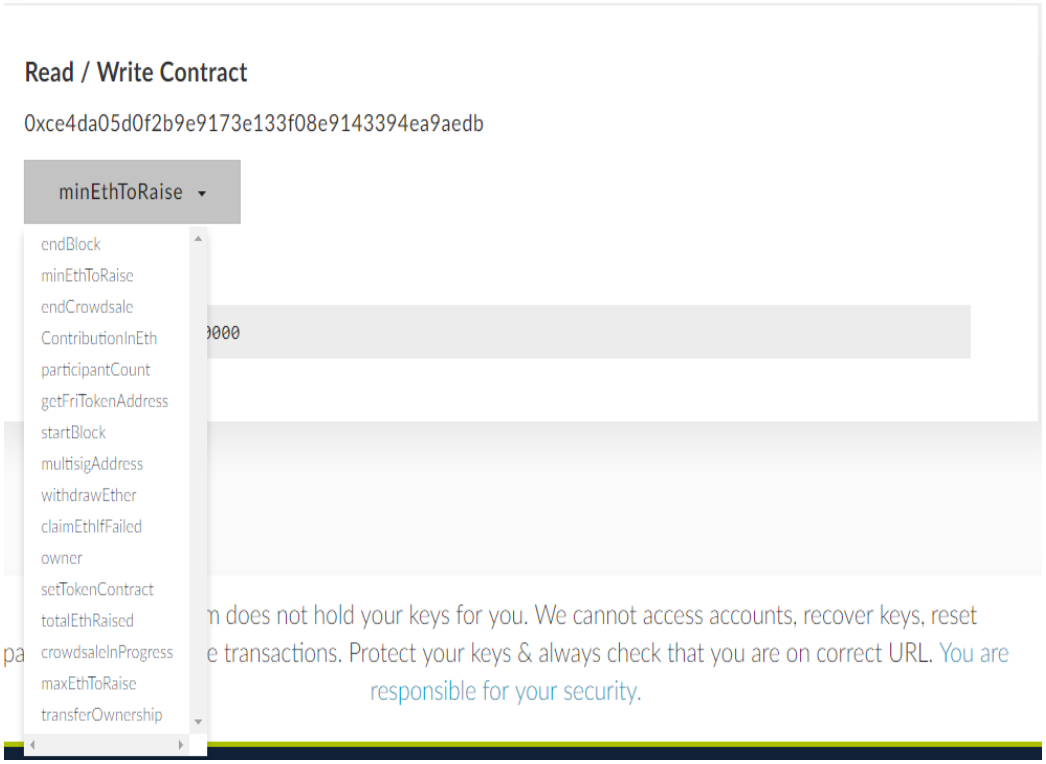
Uporabnik lahko sodeluje v prodaji z nakazovanjem poljubnega števila Ethera. V zameno dobi določeno število FRIK žetonov, odvisno od poslane količine Ethera. Ceno enega žetona sem nastavil na 10 centov.

Ob zaključku lastnik prodaje sproži akcijo za pridobitev Ethera in razdelitev žetonov vsem, ki so sodelovali.

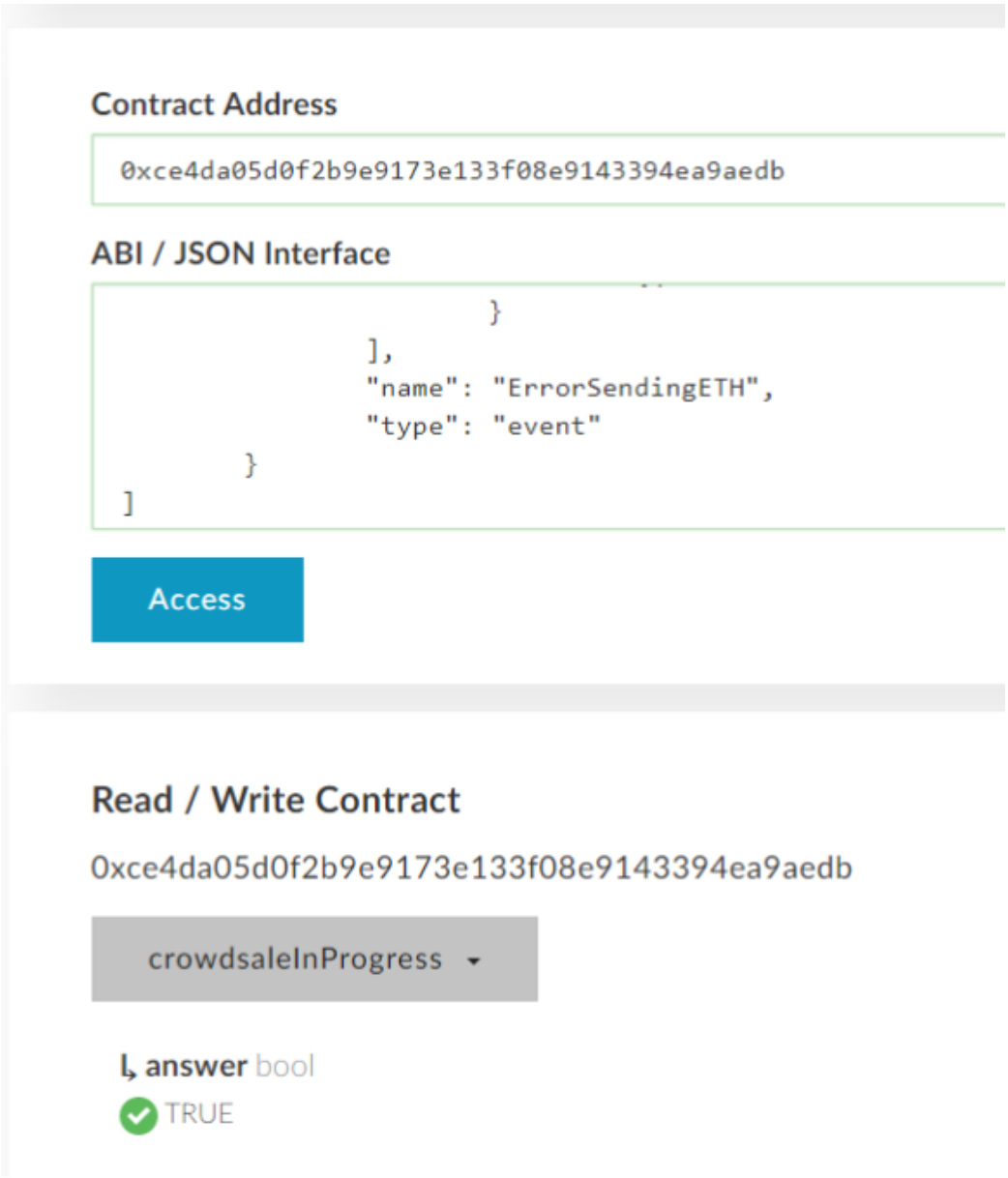


Slika 4.6: Potrjevanje transakcije.

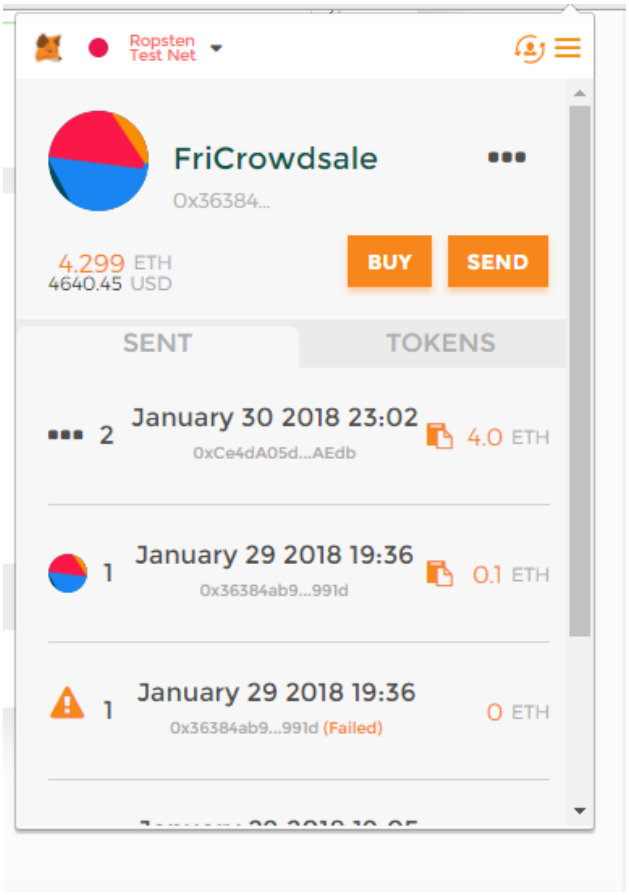
V primeru, ko prodaja ni uspešna, lastnik sproži funkcijo »claimEtherIfFailed«, ki vsem udeležencem vrne denar in označi prodajo kot neuspešno.



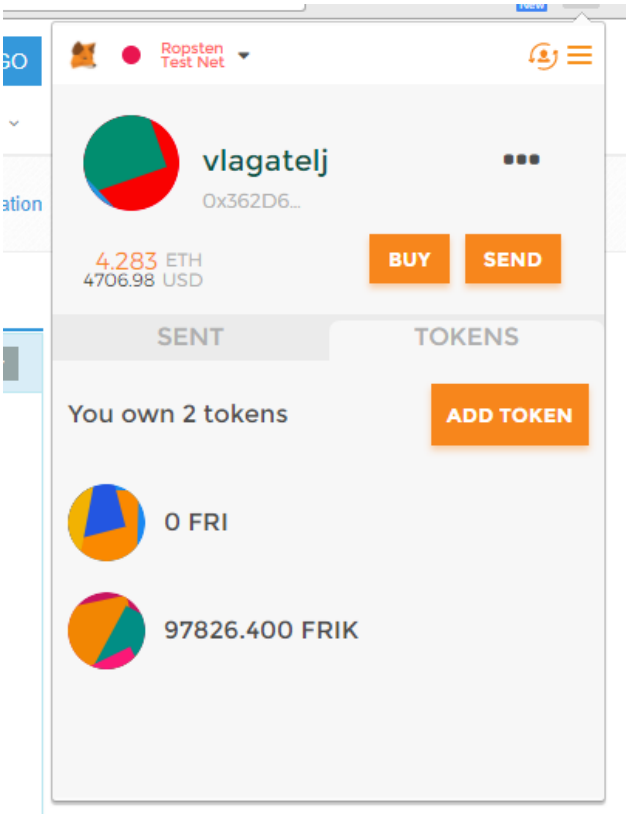
Slika 4.7: Seznam funkcij, ki jih lahko kličemo.



Slika 4.8: Preverim, če se je prodaja začela.



Slika 4.9: Pošiljanje Ethera.



Slika 4.10: Pregled FRIK žetonov, ki sem jih dobil ob zaključku prodaje.

Poglavje 5

Analiza

5.1 Čas razvoja

Za razvoj aplikacije sem vse skupaj potreboval okoli devet tednov. Približno dva tedna sem porabil za raziskovanje in učenje o Ethereumu. Nato sem kakšen teden porabil za postavitev lokalnega okolja in odpravo napak. Nekaj časa sem porabil tudi za pisanje pogodb in s spoznavanjem Solidity jezika. Okoli tri tedne sem porabil za razvoj vseh pogodb in namestitev na Ropsten omrežje ter testiranje in izvajanje celotnega procesa pridobivanja žetonov. Porabil sem precej časa samo za odpravo napak, ker je tehnologije dokaj nova. Nato sem še približno teden in pol porabil, da sem vse skupaj združil in napisal diplomsko nalogo.

5.2 Hitrost

Če se odločimo za razvoj tako, da imamo lokalno sinhronizirano vozlišče, čas sinhronizacije na računalniku s 6 GB delovnega pomnilnika, dvojedrnim procesorjem, 100 Mb povezavo ter HDD diskom traja približno 19 ur in 40 minut. Povprečni čas postavitve pogodbe v Ropsten omrežje poteka približno 15 sekund. Transakcija Ethera pa povprečno 13 sekund.

5.3 Težave pri razvoju

Največ težav je povzročala hitrost računalnika pri komuniciranju z verigo blokov, ker zahteva po lokalno nameščeni in sinhronizirani verigi blokov zaradi procesiranja vseh transakcij obremeni procesor. Izvajanje transakcij in testiranje pogodb ga dodatno obremenijo, saj je treba imeti vključeno ustvarjanje blokov za pridobivanje Ethra. Odločitev za postavitev pogodb preko portala myetherwallet.com je kvečjemu logična in nenujna, saj je vozlišče sinhronizirano na njihovem strežniku, kar omogoča lažjo postavitev. Vse skupaj je še v fazi razvoja, nič ni še čisto popolno, zato so bile težave s hrošči v programski opremi neizogibne. Ne gre zaključiti brez omembe problema, ki ga je povzročal program Mist. Zasedel je namreč celoten pomnilnik, kar je posledično pomenilo ponovni zagon računalnika, le-ta pa avtomatsko izbriše sinhronizirano vozlišče.

Poglavje 6

Sklepne ugotovitve

Ethereum veriga blokov je vsem na svetu omogočila zbiranje sredstev s tem pa tudi zagon idej posameznika, ki uporabljajo tehnologijo veriženja blokov. Poenostavila se je izdelava decentraliziranih aplikacij in izdaja žetonov, vendar je vsa tehnologija še v razvojni fazi, kar se predvsem izraža pri napakah, ki se pojavljajo pri razvoju. Razumevanje, posledično tudi reševanje takšnih napak je oteženo, saj je tehnologija razmeroma nova in ni veliko gradiva na spletu. Ethereum deluje kot svetovni računalnik in s prihodom svetovnega spleta tretje generacije je treba urediti še precej stvari, vendar smo na pravi poti k popolnoma decentraliziranemu in varnemu spletu. Razvoj takšnih projektov, decentraliziranih aplikacij, je mogoč na različnih platformah, med katerimi je Ethereum najbolj dograjen. Z izdajo svojih FRIK-žetonov bi tako lahko fakulteti omogočil, da študentom ponudi žetone, s katerimi bi lahko plačevali šolnino, izdajo potrdil, dodatne roke itd. Ob vsaki transakciji se podatki o plačilu oz. prenosu zapišejo v verigo blokov, kjer je lepo razvidno, koliko ter komu je posameznik plačal. Potreba po potrdilu o plačilu tako ni več potrebna, saj je vse zapisano v verigi, ki je transparentna. Žetone bi lahko poleg plačevanja uporabljali tudi kot sredstvo za glasovanje ali morda prijavo na izpit. Tehnologija veriženja blokov ima velik potencial, zatorej lahko z gotovostjo rečem, da je to prihodnost svetovnega spleta.

Literatura

[1] Metamask. Dosegljivo: `https://metamask.io`. [Dostopano: 1. 2. 2018].

[2] Mist. Dosegljivo: `https://myetherwallet.github.io/knowledge-base/getting-started`. [Dostopano: 13. 1. 2018].

[3] My ether wallet. Dosegljivo: `https://myetherwallet.github.io/knowledge-base/getting-started`. [Dostopano: 13. 1. 2018].

[4] Ropsten. Dosegljivo: `https://ropsten.etherscan.io`. [Dostopano: 12. 1. 2018].

[5] Solidity. Dosegljivo: `https://solidity.readthedocs.io/en/develop`. [Dostopano: 12. 1. 2018].

[6] Vitalik Buterin. Ethereum: a next generation smart contract and decentralized application platform (2013). *URL {http://ethereum. org/ethereum. html}*, 2017.

[7] Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 2013.

[8] Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.

[9] Marc Pilkington. 11 blockchain technology: principles and applications. *Research handbook on digital transformations*, page 225, 2016.

[10] Don Tapscott and Alex Tapscott. *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*. Penguin, 2016.

[11] Matjaž Gams (ured.). DIS slovarček, slovar računalniških izrazov, verzija 2.1.71. Dosegljivo: `http://dis-slovarcek.ijs.si`. [Dostopano: 1. 2. 2018].

[12] Gavin Wood. Ethereum: A secure decentralized transaction ledger, 2014.